Electronic Version
Stylesheet Version v1.1.1

Description

METHOD FOR IMPROVING PROCESSING EFFICIENCY OF PIPELINE ARCHITECTURE

BACKGROUND OF INVENTION

[0001] 1. Field of the Invention

[0002] The present invention provides a method for improving processing efficiency of pipeline architecture, and more particularity a method for determining an executing time period of a current calculation task according to a previous calculation task.

[0003] 2. Description of the Prior Art

[0004] Pipeline architecture is one of the most widely used calculation architectures for microprocessor systems. Pipeline architecture utilizes a time pulse to control a register file that can store results from executing a series of calculation tasks with a plurality of functional units, and the results are identically transmitted to a functional unit to execute a next calculation task. An advantage of pipeline ar-

chitecture is the capable of simultaneous control. Because the functional units usually execute various calculation tasks with different complexities, executing time periods of the various calculation tasks differ in length. Therefore, under a condition of utilizing pipeline architecture, a very complicated and long calculation task easily results in a wrong calculation task because data cannot synchronize at some point in time, especially under a condition of complicated data dependency between the functional units. Thus, utilizing pipeline architecture cannot suitably divide a series of complicated calculation tasks and help to simplify the complexities of simultaneous control.

[0005] Please refer to Fig.1. Fig.1 is a functional diagram of a processor 10 of pipeline architecture. The processor 10 comprises a first functional unit 12, a second functional unit 14, and a control unit 16. The first functional unit 12 is for executing a calculation task. The second functional unit 14 is for executing another calculation task.The control unit 16 is electrically connected to the first and the second functional units for generating a plurality of control signals to control the first and the second functional units 12, 14. The control unit 16 will depend upon desired calculation tasks to orderly control the first and second

functional units 12, 14 to execute calculation tasks. Simultaneously, depending upon the desired calculation tasks, the control unit 16 controls input data to the first and second functional units 12, 14 (IN1 and IN2 in Fig.1) and exports results from executing the calculation tasks of the first and second functional units 12, 14 (OUT1 and OUT2 in Fig.1).

[0006] Next, the first functional unit 12 is assumed as an Arithmetic and Logic Unit (ALU), and the second functional unit 14 is a Multiplication and Accumulation Unit (MAC). Because the MAU executes a more complicated calculation task than the ALU, the second functional unit 14 needs a longer executing time period than the first functional unit 12. For example, an executing time period of the first functional unit 12 is one instruction cycle and an executing time period of the second functional unit 14 is two instruction cycles.

[0007] Please refer to Fig.2. Fig.2 is a timing prospective view of the processor 10 executing a calculation task. Fig.2 shows when the processor 10 utilizes the control unit 16 to generate a control signal to control a functional unit for executing a calculation task, according to timing sequence levels such as fetch instruction (level F), decode (level D),

read register (level R), execution (level E1 and E2), and write back (level W). Each level as mentioned above takes one instruction cycle. Please note, parts of an executing calculation task are the parts of utilizing calculation capacities of functional units where the parts of the executing calculation tasks are only levels E1 and E2 that are in coordination with the longest executing time period of the functional unit (as the second functional unit 14) of the processor 10. If the processor 10 comprises the functional units needing longer executing time periods, parts of executing calculation tasks can increase an amount of levels depending upon requirement.

[0008] Please refer to Fig.3. Fig.3 is a timing prospective view of the processor 10 executing a series of calculation tasks. Please note, Fig.3 shows a timing of calculation task that does not consider data dependency between various calculation tasks and as an ideal status. In Fig.3, according to desired calculation tasks, the control unit 16 utilizes control signals to orderly control the first and second functional units 12,14 to execute a series of calculation tasks (as a first, a second, a third, and a fourth calculation tasks in Fig.3) wherein each adjacent calculation task differs one instruction cycle. Under such arrangement, dur-

ing a specific time period of calculation processing (as the part within dotted line in Fig.3), different calculation tasks lie in different levels. Therefore, the different calculation tasks can be executed simultaneously because of utilizing different system sources. Please note, as mentioned above, an executing time period of the first functional unit 12 is only one instruction cycle. Therefore, the prior art defines either the level E1 or level E2 to execute a real calculation task and maintains the other one in an unused status. Next, please refer to Fig.4 and Fig.5 to illustrate a condition of data dependency between different calculation tasks and a timing of the processor 10 when executing a series of calculation tasks.

[0009] Fig.4 is a timing prospective view of the processor 10 executing a calculation task of r3 = (r1 * r2) + r4. In Fig.4, r0 = r1 * r2 and the first functional unit 12 will execute a real calculation task in the level E1 (as shown within the dotted line in Fig.4) are simultaneously assumed. In Fig.4, in the first calculation task, the processor 10 will utilize the second functional unit 14 (MAU) to execute r0 = r1 * r2. After a result of the first calculation task comes out, the result will be transmitted to the first functional unit 12 (ALU) to execute the second calculation task (r3 = r0 +

r4). However, two instruction cycles are needed for an execution of the second functional unit 14 that includes the level E1 and level E2 to complete the calculation task, and the second calculation task has data dependency on the result from the first calculation task (shown as an arrow 18 in Fig.4). Thus, the second calculation task cannot be executed right after one instruction cycle of the first calculation task as shown in Fig.3. As shown in Fig.4, after two instruction cycles of the first calculation task, the second calculation task is then executed (the second calculation task stalls one instruction cycle), otherwise the result of the second calculation task will be wrong because of an incorrect data input. In the present example, the first functional unit 12 executes a real calculation task in the level E2 is initially assumed. The timing in Fig.3 will not influence a correction of the result from the calculation task because of data dependency. Therefore, as mentioned above a stall of the calculation task will not occur.

[0010] Another example is shown in Fig.5, a timing perspective view of the processor 10 executing a calculation task of r3 = (r1 +r2) * r4. In Fig.5, r0 = r1 + r2 and the first functional unit 12 executing a real calculation task in the level E2 are simultaneously assumed (as the parts within the

dotted line in Fig.5). In Fig.5, during the first calculation task, the processor 10 utilizes the first functional unit 12 (ALU) to execute a calculation of r0 = r1 + r2. After a result from the first calculation task comes out, the result is then transmitted to the second functional unit 14 (MAU) to execute the second calculation task. However, because the first functional unit 12 starts a real calculation in the level E2, the second calculation task having data dependency on the result from the first calculation task (as an arrow 20 in Fig.5) cannot be executed right after one instruction cycle of the first calculation task as shown in Fig.3. But, as shown in Fig.5, two instruction cycles after the first calculation task, the second calculation task starts to execute (the second calculation task stalls one instruction cycle), otherwise, the result of the second calculation task will be wrong because of an incorrect data input. Moreover, in the present example, the first functional unit 12 executing a real calculation task in the level E1 is initially assumed. The timing in Fig.3 will not influence a correction of the resultfrom the calculation task because of data dependency. Therefore, as mentioned above, a stall of the calculation task will not occur.

[0011] Integrated in Fig.4 and Fig.5 and as mentioned above, in

the prior art no matter whether the first functional unit 12 executes a real calculation in either the level E1 or the level E2, one stalled instruction cycle of the calculation task will possibly occur. If a calculation of r5 = abs ((r1 + r2) * r3) is considered where abs is absolute value and is executed by ALU. No matter whether the first functional unit 12 executes a real calculation task in either the level E1 or level E2, one stalled instruction cycle of the calculation task will possibly occur. In more complicated calculations, a condition of a stall will occur frequently.

[0012] In the prior art, a stall will cause a huge damage to a processing efficiency of pipeline architecture. When a stall occurs, a timing interval between two adjacent calculation tasks of writing back data is prolonged (as a cycle of time pulse controlling a register file). Under a fixed cycle of time pulse, the stall causes an increasing of the time period of the calculation task over that of the cycle of time pulse and pipeline architecture will delay one cycle of time pulse for the whole calculation resulting in a decline of processing efficiency. Therefore, a condition of Very-Long Instruction Word (VLIW) becomes obvious because calculation tasks under the VLIW condition are executed as a unit of a plurality cycle of time pulse. If a calculation task

is influenced because of the stall, the whole execution package of calculation tasks will be delayed resulting in more damage to processing efficiency of pipeline architecture.

SUMMARY OF INVENTION

[0013] It is therefore a primary objective of the claimed invention to provide a method for determining an executing time period of a current calculation task according to a previous calculation task to solve the above-mentioned problems.

[0014] According to the claimed invention, the method improves processing efficiency of pipeline architecture with a processor. The processor has a first functional unit, a second functional unit, and a control unit. The first functional unitexecutes a calculation task. The second functional unit executes another calculation task.The control unit is electrically connected to the first and the second functional units and generates a plurality of control signals to control the first and the second functional units. The method comprises: (a)executing a first calculation task with the first functional unit or the second functional unit;(b)determining an executing time period of a second calculation task with the control unit according to the

functional unit executing the first calculation task, an executing time period of the first calculation task, and whether the second calculation task depends upon a result of the first calculation task; and(c)executing the second calculation task with the first functional unit according to the executing time period of the second calculation task determined in step (b).

[0015] The method of the claimed invention will calculate a best suitable executing time period of the second calculation task according to conditions such as the type of functional unit used in a previous calculation task, an executing time period of the previous calculation task, and whether a current calculation task needs a result from the previous calculation. Then, determining the lowest possibility of a stall improves the processing efficiency of pipeline architecture.

Brief Description of Drawings

[0016] Fig.1 is a functional diagram of a processor of pipeline architecture.

[0017] Fig.2 is a timing prospective view of a processor executing a calculation task.

[0018] Fig.3 is a timing prospective view of a processor executing a series of calculation tasks.

[0019] Fig.4 is a timing prospective view of a processor executing a calculation task of r3 = (r1 * r2) + r4.

[0020] Fig.5 is a timing perspective view of a processor executing a calculation task of r3 = (r1 + r2) * r4.

[0021] Fig.6 is a diagram of a method according to the present invention for improved processing efficiency of pipeline architecture with the processor shown in Fig.1.

[0022] Fig.7 is a diagram of a method according to the best embodiment of the present invention for improved processing efficiency of pipeline architecture with the processor shown in Fig.1.

[0023] Fig.8 is a truth table of the best embodiment of the present invention.

DETAILED DESCRIPTION

[0024] Please refer to Fig.6. Fig.6 is a diagram of a method according to the present invention for improved processing efficiency of pipeline architecture with the processor 10 shown in Fig.1. The method comprises:

[0025] Step 22: Start;

[0026] Step 24: Executing a first calculation task with the first functional unit 12 or the second functional unit 14;

[0027] Step 26: Determining an executing time period of a sec-

ond calculation task with the control unit 16 according to the functional unit executing the first calculation task, an executing time period of the first calculation task, and whether the second calculation task depends upon a result of the first calculation task;

[0028] Step 28: Executing the second calculation task with the first functional unit 12 according to the executing time period of the second calculation task determined in step 26; and

[0029] Step 30: End.

[0030] Different from the method of executing calculation tasks in levels (also called as static pipeline stage control) of the prior art, the present invention executing a current calculation task (as the second execution task) will in advance judge that the previous calculation task (as the first calculation task) is executed by which functional unit, the previous calculation task is really executed by which level, and whether the current calculation task has any data dependency on the previous calculation task. Then, according to a result from judging the above-mentioned conditions, the most suitable level to execute the real calculation task is calculated for the current calculation task. In contrast to the prior art, the present method is a dynamic

pipeline stage control.

[0031] Next, please refer Fig.7 to describe detailed execution steps of one of the best embodiments of the present invention. Fig.7 is a diagram of a method for improved processing efficiency of pipeline architecture with the processor 10 as shown in Fig.1. In the present embodiment, the first functional unit 12 is ALU and the second functional unit 14 is a MAU are assumed. Because the MAU executes a more complicated calculation task, the second function unit 14 needs an executing time period longer than an executing time period of functional unit 12. The executing time period of the first functional unit 12 is one instruction cycle, and the executing time period of the second functional unit 14 is two instruction cycles. Please note, the control unit 16 of the present embodiment generates a plurality of control signals to orderly control the first and the second functional units 12, 14 as in a first time period (as the first calculation unit in the level E1) and a second time period (as the first calculation unit in the level E2) executing the first calculation task, and in a second time period (as the second calculation unit in the level E1) and a third time period (as the second calculation unit in the level E2) executing the second calculation task. The

lengths of the first, second, and third time periods are all equal to one instruction cycle, and the first, second, and third time periods are non-overlapping with one another, and the second time period is later than the first time period and the third time period is later than the second time period.

[0032] As shown in Fig.7, the method comprises (please note, the following steps occur after the previous step 24):Step 32: Utilizing the control unit 16 to check whether the first calculation task is executed by the first functional unit or not. If yes, Step 34 is executed. If no (the first calculation task is executed by the second functional unit 14), Step 36 is executed.

[0033] Step 34: Utilizing the control unit 16 to check whether the first calculation task is executed in the second time period or not. If yes, Step 40 is executed. If no, Step 36 is executed.

[0034] Step 36: Utilizing the control unit 16 to check whether the second calculation task has data dependency on a result from the first calculation task or not. If yes, Step 40 is executed. If no, Step 38 is executed.

[0035] Step 38: Utilizing the control unit 16 to control the first functional unit 12 to execute the second calculation task

in the second time period.

[0036] Step 40: Utilizing the control unit 16 to control the first functional unit 12 to execute the second calculation task in the third time period.

[0037] Consider a condition of the second functional unit 14 executing the first calculation task in the first and second time periods and the first functional unit 12 executing the second calculation task. If the second calculation task does not need the result from the first calculation task, the first functional unit 12 is controlled by the control unit 16 to execute the second calculation task in the second time period. If the second calculation task needs the result from the first calculation task, the first functional unit 12 is controlled by the control unit 16 to execute the second calculation task in the third time period.

[0038] Now consider a condition where the first functional unit 12 executes the first calculation task in the first time period and also executes the second calculation task. If the second calculation task does not need the result from the first calculation task, the first functional unit 12 is controlled by the control unit 16 to execute the second calculation task in the second time period. If the second calculation task needs the result from the first calculation task,

the first functional unit 12 is controlled by the control unit 16 to execute the second calculation task in the third time period.

[0039] Lastly, consider a condition where the first functional unit 12 executes the first calculation task in the second time period and also executes the second calculation task. The first functional unit 12 is controlled by the control unit 16 to execute the second calculation task in the third time period. The above-mentioned method can be described in Fig.8. Fig.8 is a truth table of the best embodiment of the present invention wherein a logic value "0" in a column means no, a logic value "1" in a column means yes, and "x" means irrelative.

[0040] Next, different calculation processes are described about how to reach a best calculation efficiency of the best embodiment of the present invention. There are three kinds of calculation methods. The first method is to assume that the first functional unit 12 will execute a real calculation task in the level E1 as the method of static pipeline stage control in the prior art. The second method is to assume that the first functional unit 12 will execute a real calculation task in the level E2 as the method of static pipeline stage control in the prior art. The third method is the

method described in the best embodiment. Please note, in the third method the first functional unit 12 initially executes a real calculation in the level E1 and executes a corresponding switch according to conditions.

[0041] First, an example as mentioned above, r5 = abs ((r1 + r2)*r3). There are three calculation tasks needed in this calculation process. The first calculation task r0 = r1 + r2 is completed by utilizing the ALU, the second calculation task r4 = r0 * r3 is completely by utilizing the MAC, and the third calculation task r5 = abs (r4) is completed by utilizing the ALU. Under an inference from a timing perspective view of above-mentioned Figs. 4 and 5, results can come out as: utilizing the first method to execute calculation tasks, a stall of one instruction cycle occurs between the second and the third calculation tasks; utilizing the second method to execute calculation tasks, a stall of one instruction cycle occurs between the first and the second calculation tasks; and utilizing the method of the present invention to execute calculation, no stall occurs.

[0042] Moreover, an example of a 4-byte integer array searching a byte integer with (X * X) integer index back from an initial site is described. A site "A" of the byte integer can be described as an equation of $A = 4X^2 + I$ wherein I is the

initial site. There are three calculation tasks needed in this calculation process. The first calculation task B = X * X is completed by utilizing the MAU, the second calculation task C = 4B is completely by utilizing the ALU (moving B two bytes in the left), and the third calculation task A = C + I is completed by utilizing ALU. Under an inference from a timing perspective view of above-mentioned Figs. 4 and 5, results can come out as: utilizing the first method to execute calculation tasks, a stall of one instruction cycle occurs between the first and the second calculation tasks; utilizing the second method to execute calculation tasks, no stall occurs; and utilizing the method of the present invention to execute calculation tasks, no stall occurs.

[0043] Finally, an example of e = (a + b) * (c + d) is described. There are three calculation tasks needed in this calculation process. The first calculation task f = a +b is completed by utilizing the ALU, the second calculation task g = c + d is completely by utilizing the ALU, and the third calculation task e = f * g is completed by utilizing the MAU. Under an inference from a timing perspective view of above-mentioned Figs. 4 and 5, results can come out as: utilizing the first method to execute calculation tasks, no stall occurs; utilizing the second method to execute

calculation tasks, a stall of one instruction cycle occurs between the second and the third calculation tasks; and utilizing the method of the present invention to execute calculation tasks, no stall occurs.

[0044] In contrast to the method of static pipeline stage control of the prior art, the method of dynamic pipeline stage control of the present invention will calculate a best suitable time period for executing the second calculation task according to conditions such as the type of functional unit used by a previous calculation task, an executing time period of the previous calculation task, and whether a current calculation task needs a result from the previous calculation. Then, determining the lowest possibility of a stall improves processing efficiency of pipeline architecture.

[0045] Those skilled in the art will readily observe that numerous modification and alternations of the present invention method may be made while retaining the teachings of the invention. Accordingly, the above disclosure should be construed as limited only by the metes and bounds of the appended claims.